

# REDUCING THE COST OF GROUND SYSTEMS & OPERATIONS THROUGH SOFTWARE RE-USE

Chris Gilbert  
Astrium GmbH, Space Infrastructure  
Business Development, Operations  
Christopher.Gilbert@astrium-space.com

Wim van Leeuwen  
European Space Agency  
Head of Ground Infrastructure Implementation  
Wim.van.Leeuwen@esa.int

## ABSTRACT

The International Space Station (ISS) is the largest single international undertaking in space and is collectively managed by the space agencies of the USA, Europe, Japan, Russia and Canada. The principal elements of the European contribution are the Columbus pressurized module and the advanced transfer vehicle (ATV). On-orbit activities in the Columbus module are supported by an extensive ground command and control infrastructure serving Columbus system operations and the user community needs. A central feature of this infrastructure is the re-use of computer software developed initially to support the integration and checkout of the Columbus module. Known as CGS (Columbus Ground Software), this software was conceived as an end-to-end data system from which all the Columbus development, simulation and software test facilities have been derived. Key to the design of CGS is a single Mission Data Base (MDB) as the central engineering knowledge repository. CGS software is re-used in the Monitoring & Control Subsystem (MCS), which forms the core of the Columbus operations control center (COL-CC). This paper reviews the architecture of the CGS software, and its application to the Columbus ground segment during development and subsequently for operations.

## 1. INTRODUCTION

The launch of the Columbus module, currently planned for October 2004, and its subsequent installation on the International Space Station, will mark the end of a long period of development and preparation of the module and its payloads. It will also mark the beginning of a period of completely new activities with new challenges - on-orbit operations. Under contract to ESA, the European Space Agency, a comprehensive ground system is being established in preparation for the operational phase. The Columbus orbital laboratory is the tip of an extensive control and communications hierarchical infrastructure that integrates the space segment operations needs and the payload operation needs of the user community (Fig. 1).

The resulting network provides appropriate access to the infrastructure at various levels of involvement. The network is distributed across many European countries, enabling scientists from participating ESA member states to maintain real-time contact with their experiments. There are several reasons for implementing a common software

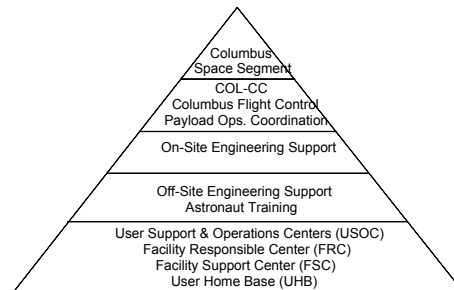


Figure 1: Columbus Operations Hierarchy

operations concept to support this diverse user community, in particular to guarantee consistency between development activities and operations. This is achieved by ensuring that data qualified and verified in development is re-used as the basis for mission control. An equally relevant reason is to minimize cost over the project life cycle. Cost reductions arise primarily by avoiding duplicated development efforts, but also from:

- re-use of test procedures, displays and models between facilities,
- reduced training costs for staff operating the different facilities, and
- by taking ground software off the critical path of the schedule, thereby gaining schedule margin and reducing program risk.

## 2. CGS IN THE DEVELOPMENT PHASE

**2.1. CGS Software Architecture:** The basic modular architecture of CGS, showing the main functional sectors as they were originally conceived, is depicted schematically

in Fig. 2. This modular approach with its building blocks on a common foundation of infrastructure and data base has enabled the re-use of the CGS from initial application as EGSE to configure all other Columbus ground facilities, including a payload verification facility, crew trainers, and simulators for compatibility tests with other ISS elements. In addition, this capability is exploited in the mission control system, all eleven Columbus payload elements, and for the Columbus simulator used to support the operations preparation.

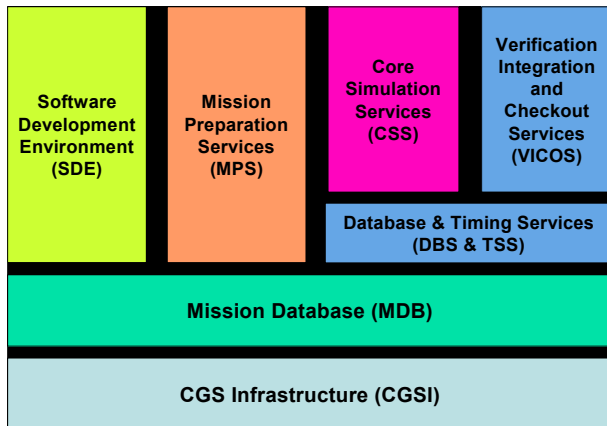


Figure 2. CGS schematic architecture

Underlying the applications modules is a commercial operating platform available on Linux or Sun Solaris running Oracle. A number of specialised tools have been developed to manage the applications as the environment has evolved.

**2.1.1. The CGS Infrastructure** provides an essentially COTS-based hardware and software platform tailored to support specific applications standards such as PUS and ECSS. The current version runs on SUN / Solaris workstations or on Linux / PC.

**2.1.2. The Mission Data Base** runs on Oracle 8/9 and Unix, optionally on Linux, providing a stable and reliable platform. The main functions of the MDB are:

- centralized data and data definitions, storage and distribution to all facilities;
- overall consistency checking, and
- configuration control.

In short, the MDB provides the engineering knowledge repository with which the simulation, verification and checkout activities are supported. Fig. 3 shows the basic architecture of the MDB:

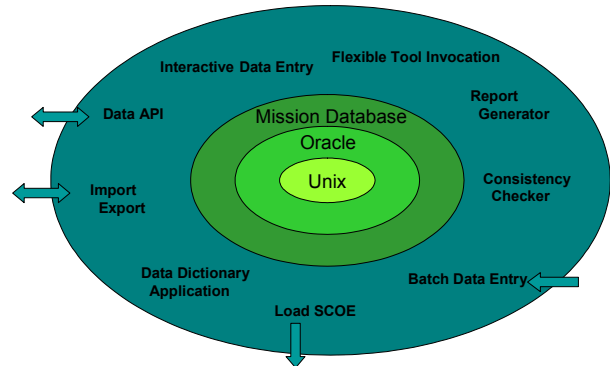


Figure 3: Mission Data Base Architecture

The data-base is populated at the lowest level by end items. These are real data entities, and include: all measurement and command end items, and their attributes; calibration and limit definitions; telecommand and telemetry packet layouts; synoptic display definitions; automated checkout sequences; interactive sequences; test results (log events, messages, raw data). Its architecture provides a number of internal tools enabling data entry, reporting, preparation of checkout and simulated sessions, off-line generation of onboard and flight software images. In particular, all end items are under the control of an automatic configuration function. This function is of central importance to the applicability of CGS throughout the project lifecycle. The MDB installation in the NASA MBF currently supports over 1 million end items. The Columbus orbital laboratory MDB will contain around 35,000 end items. Fig. 4 shows an example of the end item tree structure. The tree is divided into a system tree and user trees. Responsibility for maintaining each user tree resides with the appropriate user. He has considerable freedom in defining his end items and only needs to conform to path and interface definitions. Logical groups of end items are collected into configuration control units (CCU). The combination of CCU and end items enables the parallel configuration control of consistent configurations (i.e. Engineering Model and Flight model).

System Tree and User Tree

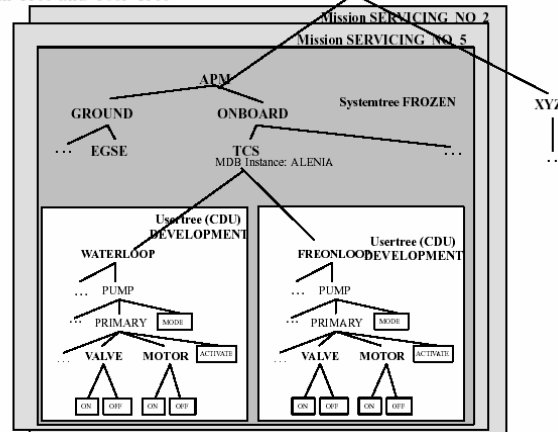


Figure 4: End Item Tree

Interactive data entry is used for online data entry and retrieval sessions by a user at a work station. A utility for batch data entry simplifies the procedure when entering large volumes of data in non-interactive mode. Batch data entry is also used for data exchange with external databases. Before formal release, the database contents (e.g. flight configuration data) undergo several consistency checks to ensure overall data integrity.

CGS is itself a distributed software system. This means that the database, the human computer interface (HCI) software and the real-time test execution kernel are separate software modules which can be run on dedicated computer systems. This architecture enables the overall performance of CGS to be scaleable simply by adding or removing computers as appropriate. The configuration database concept enables parallel, distributed data development at different geographic locations, and for data exchange between these locations. Subsequent integration of data at the system integrator's site is a built-in capability of CGS. This is achieved by an end-item ownership concept and ensured by restricted security and access mechanisms.

2.2. Design and Development Phase: The life cycle support provided by CGS during the design and development phase consists of:

Engineering support for requirements (structured analysis and design method) and interface control documents;

Software design support including OOD Methodology for architectural design, detailed design, and program design of application software;

Software coding and testing support including syntax sensitive editing for Ada and C source code and software compiler systems;

additional, optional software development tools which can be integrated through a standard generic compiler and tool interface;

Generation of automated procedures written in the User Control Language (UCL) supported by dedicated editors and compilers for execution in the CGS based ground system or in the onboard system;

Generation of active (synoptic) displays for use in the CGS based ground system or onboard the target system,

Generation of simulation models for execution in the CGS based ground system;

Software and document handling, and configuration management.

2.3. CGS Support for Integration, Testing and Qualification

2.3.1. The Simulation Function Block:

Simulation plays a special role throughout the system life cycle. The simulation function block simulates missing hardware and software components during the early integration phase of target systems.

During testing in the conceptual phases, it enables the early check of alternative solutions and supports rapid prototyping. During integration, components and subsystems may be tested by simulation of missing components or by simulation of the remaining system. While hardware is in production, pre-programmable automatic testing on the component level, subsystem level, and system level reduces the effort for qualification. During the operational phase, simulation enables revision in case of unplanned or unforeseen events. Simulation additionally supports fault detection, isolation, and recovery.

The simulation environment enables the definition of operator interfaces with active displays, menus, and switches. By linkage of these interfaces with the simulated functions complete simulators for training purposes are created with the system features already defined and realized during development. Cooperation with features of the test configuration, e.g. the active graphical displays, enables the generation of training facilities with simulated real response capabilities. Virtual operation boards can also be realized.

The creation of a simulation model is supported by an easy to use graphical model development environment. Predefined model function libraries are provided. User specific libraries may be created to include additional model functions implemented either by decision tables or Ada or C code.

Executable simulation code can be generated automatically, independent of the model abstraction level. The simulation model source and the executable simulation code are stored in the MDB.

Through the graphical model source, the user can monitor and control simulation during execution. The events and results of the model execution are stored in the test result database for evaluation using the CGS test software.

The adaptation of the generic CGS simulation to the target system's I/O-Controllers is performed by special application software in the Command and Measurement Adaptation System (CMAS) using CGS API (Application Programming Interface). This results in the capability for hardware-in-the-loop tests.

2.3.2. The Test Function block:

The test function block supports test operations and automatic testing and monitoring of the unit under test. Testing of components, subsystems, or the target system, meet several goals:

- demonstration and verification of functionality
- fit check to other items
- system test
- certification of fulfillment of requirements and customer acceptance

The facilities for testing hardware and software components require software to operate those facilities, to specify the tests, to perform the tests and to analyze the test results. The implementation of a large, distributed test environment with special test node computers allows for thousands of active end items to be involved in real time tests. The integrated language system avoids the necessity for programming tests, enabling the operators to focus fully on the test itself.

The test function block of CGS is itself a distributed software system. In that module, the database, the HCI software, and the real-time test execution kernel run on separate computers. Due to this architecture the overall performance of the test system is scaleable by simply adding or removing computers.

The CGS test function block is driven by the test definitions stored in the CGS Master Database. It provides a generic data and control interface to the UUT and all services required for real-time data processing.

Data stored in the CGS Master Database consists of the command and measurement list, ground active displays (synoptics), automated procedures written in User Control Language (UCL), and command sequences written in High Level Command Language (HLCL) to interactively control the test system.

The command and measurement list is entered into the CGS Master Database either through the interactive user interface or by way of the batch data entry capability.

The active displays are generated using the CGS Ground Window Definition Utility. This utility is based on the commercial tool "Dataviews" but it has been customized to the CGS needs and concepts.

The test system may be controlled in various ways:

by interactive control via graphical user interface (point and click) including previously generated active displays;  
or

by interactive control via HLCL keyboard commands or HLCL command sequences; or

by automatic control via compiled ground automated procedures written in UCL.

HLCL and UCL have the same syntax, however UCL cannot be used in an interactive way whereas HLCL can. HLCL is an interpretative command language.

The events and results of a test execution session are stored in the test result database. Here they are available for parallel or later result evaluation using the CGS Test Evaluation Software (TES).

The CGS Test Evaluation Software (TES) provides all services to evaluate data generated and stored in the test result database. It provides built-in capabilities for data selection, calibration, and presentation, and it serves to

generate data sets compatible with the MS Excel spreadsheet format.

For a distributed system such as CGS, it is necessary that all the computer clocks are highly synchronized. A CGS time service synchronizes the local computer clocks with a master clock. The master clock can be either an external master time unit or an internal selected computer clock (e.g. the local computer clock of the master test processor). In addition, the CGS time services provide and maintain a second time, the Simulated Mission Time (SMT) an artificial onboard or mission or test session time.

It becomes evident that the database of qualified and verified end items is an important source of engineering data which could with advantage be exploited in the operations phase. Fig. 5 illustrates the philosophy:

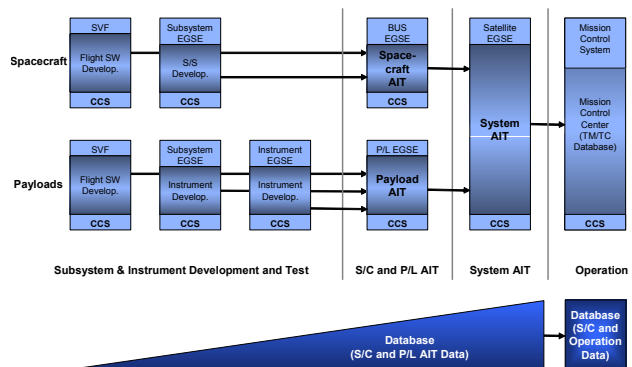


Figure 5: Vertical Integration Concept

### 3. CGS IN THE OPERATIONAL PHASE

As mentioned in the introduction, the Columbus ground segment comprises ground facilities providing technical support to Columbus operations, and a European-wide user network. Fig. 6 provides an overview of the locations of these facilities.



Figure 6: Location of Columbus Ground Facilities and User Centers

**3.1. The Scope of the Columbus Ground Network:** The focal point of this network is the Columbus Control Center (COL-CC), an ESA facility hosted by the German Space Operations Center (GSOC) in southern Germany. The software kernel of the COL-CC, the Monitoring & Control Subsystem (MCS), is based on CGS, its functionality expanded for control centre operations. The TQVS (Training, Qualification & Validation Subsystem) flight system simulator – part of COL-CC and used for ground operator training – will also be based on CGS. In addition, the following ground support systems use CGS software:

- Columbus Electrical Ground Support Equipment (EGSE),
- Columbus software integration and validation facilities (SITE),
- Crew trainers (TRE & TRU),
- Columbus flight system simulator at the NASA Software Development and Integration Laboratory (Software Verification Facility),
- the Columbus facility for payload compatibility tests (RLTF).

In its entirety, as shown in Fig. 7, the ground system network also includes the ATV Control Center.

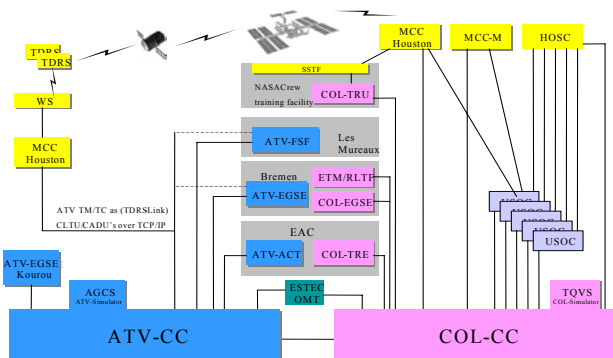


Figure 7: Columbus Ground Network Architecture

CGS software is also used by NASA at its ISS Mission Build Facility in Houston, TX, and by ESA for DMS-R, ATV and other satellite EGSE systems

**3.2. CGS Functionality for Columbus-CC:** Re-using the CGS-based MCS in COL-CC, together with the qualified Mission Data Base from the Columbus module checkout activities substantially reduces the risks of incompatibility between the ground and flight systems. The expanded CGS architecture providing the necessary Monitoring & Control functions is shown in Fig. 8.

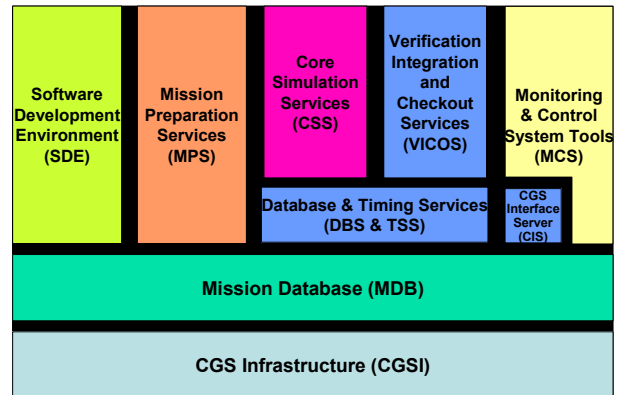


Figure 8: Extended CGS/MCS Architecture

CGS contains all the necessary components to support the operational phase of the space segment, comprising:

- the software development environment for controlled development of onboard software (e.g. flight automated procedures),
- simulation for software verification before transmission to the flight system, and
- the check-out function as a basis for command and monitoring of the flight system.
- infrastructure for remote control is also provided in CGS-based systems.

The COL-CC will operate Columbus in close cooperation with the Space Station Control Center, Houston (MCC-H) and the Payload Operations & Integration Center (POIC) at the Huntsville Operations Support Center (HOSC). It will also provide support functions for payload operations including:

- routing of telecommands from the payload operations site to the Columbus module,
- distributing support data to the USOCs,
- archiving low and medium rate telemetry, telecommands, audio and video data, and
- supporting local video rooms.

Fig. 9 shows a schematic overview of the Columbus-CC design.

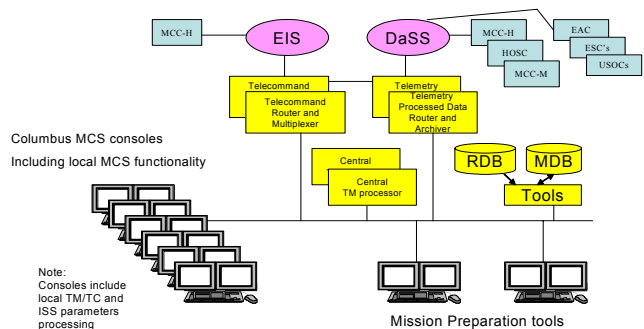


Figure 9: Columbus-CC schematic overview

3.3. Distributing the MCS Software to the Users: The exploitation of the Columbus space segment can only be successful if all members of the user community are assured of optimum access to their experiment. The next logical step, therefore, is to provide a common monitoring & control software system in each User Center. This variant, known as Columbus Distributed MCS (CD-MCS) is essentially an individual CC console at each User Center, and will ensure a seamless operation and data exchange between each User Center and the COL-CC. This approach has a number of inherent advantages. It provides for:

- Compatibility with COL-CC operations, interfaces and data presentation;
- Common and centralized maintenance with other Columbus ground support facilities (COL-CC, Crew trainers, EGSE, etc),
- Overall MDB based configuration control and unique data definitions (TC, TM calibration curves, ISS processed data, etc),
- Direct use of ISS processed parameters and related data definitions from the Columbus Mission Database (including mapping between DaSS naming "UMI" and NASA PUI's),
- Cost-effective solution

3.4. The USOC Network: The User Support and Operations Centers currently planned match the payload facilities and experiments supported by ESA and the member states. The Centers operate at different levels according to the tasks they undertake. A Facility Responsible Center (FRC) has full responsibility for overall management of a specific payload facility in Columbus. A Facility Support Center (FSC) provides support for certain functions of an ESA-developed multi-user facility. Ultimately, an individual user will have access to the ground system from his User Home Base (UHB). The pressurized payloads and primary USOC assignments are as follows:

Biolab, operated by MUSC in Cologne as the FRC, with support by BIOTESC in Zurich;

Fluid Science Facility (FSL), operated by MARS in Naples as the FRC, with support by IDR in Madrid;

European Physiology Module (EPM), operated by CADMOS in Toulouse as the FRC with support by DAMEC in Copenhagen;

European Drawer Rack (EDR), operated by the ERASMUS center at Estec in Noordwijk as the FRC, with support by the Belgian USOC and the DUC;

Materials Science Laboratory (MSL), furnaces operated by MUSC and CADMOS as FRC with support by each other;

European Modular Cultivation System (EMCS) operated by the Norwegian USOC in Trondheim.

3.5. CD-MCS Installation: The software/hardware package being developed for the USOCS is shown schematically in Fig. 10.

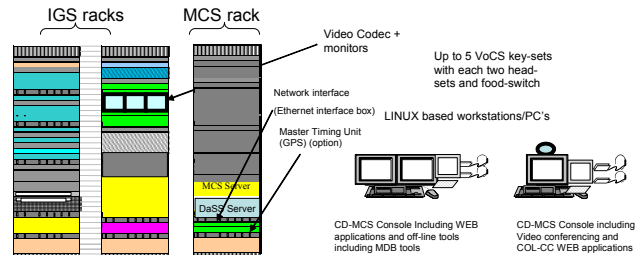


Figure 10: CD-MCS package for USOC Installation

The CD-MCS provides the following standard MCS functions:

- Telemetry processing (acquisition, calibration, monitoring, exception handling,
- Telecommands (from different sources e.g. manual command stack, synoptic displays, automated procedures, keyboard inputs (HLCL), SAS (special application software),
- ISS, GS and COL-CC processed parameter handling (receiving and delivering processed parameters),
- MMI including synoptic displays, alphanumeric displays, exception displays etc,
- Automated procedures executer including debugger,
- Archiving and play-back,
- Operations preparation tools (for procedures, command stacks editor, alphanumeric displays etc.),
- Evaluation tools (dumps, histograms and "MS-Office" compatible file outputs),
- MDB and related tools (including Payload-DB, an MDB-derived tool),
- P/L specific additions (MDB additional type, TES command update and private header support),
- On-board time synchronization with COL-CC,
- Step back processing,
- various CGS Payload adaptations.

#### 4. CONCLUDING REMARKS

The widespread implementation of the Columbus Ground Software system and its unique Mission Data Base forms a rational basis for cost-effective mission and payload operations over the extended lifetime of the International Space Station. The high safety standards necessary for safe and productive exploitation of the ISS are an inherent part of this approach since all relevant data, once qualified and verified, are securely protected and used as a common source of engineering information throughout the ground and flight segments. Over the lifetime of Columbus, planned maintenance and upgrade activities will provide regular productivity improvements and enable utilization goals to be realised. Combining this superior technical performance with reduced implementation costs is the key

to future progress in this and many other fields of endeavour.

Fig. 11 symbolizes this "life cycle" approach to reducing costs.

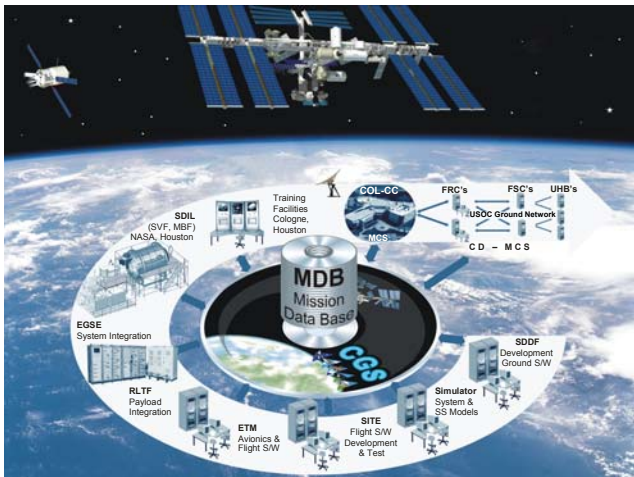


Figure 11: The CGS product life cycle support concept

## 5. BIBLIOGRAPHY

- [1] Hummel, J. & Barth, F.-H. *Columbus Ground Software for Development & Operation of the ISS*. Technical note.
- [2] van Leeuwen, W. *ESA's Manned Spaceflight Ground Segment*. ESA Technical Note, December 2001
- [3] Chesson, R. *The ISS Operations & Exploitation Program*. ESA Bulletin No. 110, May 2002
- [4] Graf, J. *ISS Operations Preparation & Exploitation*. Unpublished ESA Technical Note, September 2002
- [5] van Leeuwen, W. *CD-MCS for USOC Outfitting*. Unpublished ESA technical note, April 2003.